Vysoká škola báňská – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Bitové a logické operace v číslicové technice

Bitwise and Logical Operations in Digital Circuits

2016 Tomáš Schűhler

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání bakalářské práce

Student: **Tomáš Schűhler**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R059 Mobilní technologie

Téma: Bitové a logické operace v číslicové technice
Bitwise and Logical Operations in Digital Circuits

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Popište bitové operace používané v číslicové technice.
2. Popište logické operace používané v číslicové technice.
3. Vysvětlete rozdíly mezi bitovými a logickými operacemi.
4. Popište možnosti realizace jednotlivých bitových a logických operací pomocí hardwarových prostředků (hradla, procery, VHDL) a softwarových prostředků (C, C++, Java)
4. Vytvořte interaktivní animaci/simulaci detailně popisující chování bitových i logických operací.
5. K animaci/simulaci vytvořte studijní opory pro studenty a pedagogy. V oporách shrňte základní teorii k bitovým a logickým operacím a popište jak používat Vámi vytvořené animace/simulace.

Pro vypracování závěrečné práce bude použit typografický systém LaTeX.

Seznam doporučené odborné literatury:

PETZOLD, Charles, Michael J SCHEARER a Frank THORNTON. *Code: the hidden language of computer hardware and software*. [Pbk. ed.]. Redmond, Wash: Microsoft Press, 2000, ix, 413 p. ISBN 978-073-5611-313.

KULISCH, Ulrich, Michael J SCHEARER a Frank THORNTON. *Advanced arithmetic for the digital computer: design of arithmetic units.* [Pbk. ed.]. Wien: Springer, c2002, xii, 141 s. ISBN 32-118-3870-8.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Miroslav Bureš**

Datum zadání: 01.09.2014

Datum odevzdání: 15.07.2016

doc. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky 1D16 OSTRAVA

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: 15. *července 2016*

podpis studenta

Rád bych tímto poděkoval panu Ing. Miroslavu Burešovi za pomoc. Dále bych rád poděkoval rodině a profesorům, kteří pomohli ke vzniku této práce.

Abstrakt

Tato práce vznikla za účelem přiblížení bitových a logických operací v číslicové technice. Popisuje a porovnává jejich rozdíly pro snadnější pochopení dané problematiky, včetně ukázek zapojení pomocí hardwarových prostředků a použití v programování. Její součástí jsou interaktivní schémata na webových stránkách, demonstrující chování obou druhů operací. Práce obsahuje i studijní opory pro studenty a pedagogy, které popisují, jak se schématy pracovat.

Klíčová slova: číslicová technika, výukový materiál, integrované obvody, fpga, hradla

Abstract

This work was created in order to closely introduce bitwise and logical operations in digital circuits. It describes and compares their differences between bitwise nad logical operations for a purpose of easier understanding of these matters including examples of interconnections of hardware components and their utilization in programming. Part of the work consists of interactive schemes on web pages which demonstrate behavior of the both kinds of operators. The work also comprises a study support for students and teachers describing how to operate with schemes.

Keywords: digital circuits, educational material, integral circuits, fpga, gates

Seznam použitých zkratek a symbolů

2D	–	Two dimensional
ALU	–	Arithmetic logic unit
CLB	–	Configurable Logic Block
CSS	–	Cascading Style Sheets
ESD	–	Electrostatic Discharge
FPGA	–	Field Programmable Gate Array
HTML	–	Hyper Text Markup Language
IE	–	Internet Explorer
IEEE	–	Institute of Electrical and Electronics Engineers
JS	–	Java Script
LED	–	Light Emitting Diodes
LMS	–	Learning Management System
MSB	–	Most Significant Bit
LSB	–	Least Significant Bit
SGML	–	Standard Generalized Markup Language
STD	–	Science and technology Internet Standard
SVG	–	Scalable Vector Graphics
USB	–	Universal Serial Bus
VHDL	–	VHSIC Hardware Description Language
VHSIC	–	Very High Speed Integrated Circuit
XHTML	–	Extensible Hyper Text Markup Language
XML	–	Extensible Markup Language

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Tato práce se zabývá základními principy fungování logických a bitových operací. Počítače jsou složeny z komponent, které se výrábějí od těch nejjednodušších, jako jsou logická hradla, čítače, paměti a AD/DA převodníky, až po složitější mikrokontroléry nebo programovatelná hradlová pole. Tyto komponenty vycházejí z logických obvodů kterým se práce věnuje.

První částí je tento úvod samotný. Druhá se zabývá teoretickými základy, které jsou nezbytným podkladem pro následné porozumění probírané tématiky. Na konci druhé kapitoly je popsán rozdíl mezi logickými a bitovými operacemi. V čem se odlišují a proč je nemůžeme zaměnit. Třetí část obsahuje popis a bloková schémata logických operací, která doplňují i jejich pravdivostní tabulky.

Ve třetí části se zaměřuji na realizaci pomocí softwarových a hardwarových prostředků. Pro softwarové řešení je připravena ukázka implementace v C/C++, Javě a Pascalu. V případě hardwaru byly operace domenstrovány na procesoru i486 a vyšším, dále pak byla použita hradlová stavebnice Modular RC 2000, kde je ukázka reálného zapojení základních dostupných hradel, která probíhala v učebně. Na FPGA kitu Digilent Basys2 je pak pomocí schémat a VHDL reprezentována funkčnost hradel. Čtvrtá se zabývá animacemi přibližující logické a bitové operace na webových stránkách. Poslední kapitola popisuje, jak s těmito animacemi pracovat.

2 Teoretické základy

2.1 Číslicová technika

Používá se často při zpracování a výpočtu dat v řídicích systémech, komunikacích a při různých měřeních. Mnoho úkolů dříve řešených pomocí analogových systémů je v současné době prováděno digitálně. V číslicovém systému fyzikální veličiny nebo signály jsou reprezentovány diskrétními hodnotami. Zatímco v analogových systémech se mohou fyzikální veličiny a signály průběžně měnit v závislosti na stanoveném rozsahu. Číslicový systém bývá navržen podle teorie digitálního logického obvodu. Každý číslicový systém přístrojů pracuje s digitálními signály, které nabývají pouze dvou hodnot, typicky 0 nebo 1.[1, 3]

2.2 Booleova algebra

Vypracoval ji irský matematik George Boole jako aplikaci matematiky v oblasti logiky. Je základním matematickým nástrojem pro analýzu a syntézu logických obvodů ve všech typech. Návrh a popis hardwaru se provádí pomocí Booleovské funkce a konečného automatu. V praxi je poprvé použil Claude Shannon až o hodně let později. Tyto nástroje syntézy se používají k popisu digitálních soustav a jsou vhodnou formou pro jejich realizaci. Je důležité znát formát dat, algoritmus zpracování a také terminologii. Své praktické uplatnění si našla až osmdesát let po svém vzniku a to jako vhodný matematický aparát pro řešení logických obvodů. Využívá úsporný algebraický zápis logických vztahů. Pomocí zjednodušení a různých úprav logických funkcí Booleova algebra utváří hospodárný a minimalizační návrh.[1, 2]

Obrázek 2.1: Irský matematik George Boole[4]

2.3 Boolovská funkce

Definuje se obdobně jako matematická funkce. Její vstupní proměnné nabývají logické hodnoty. Je základním popisem kombinačního logického obvodu a jsou s ní také odvozeny všechny následné kroky pro navrhování obvodů. Rozděluje se na úplnou a neúplnou booleovskou funkci, kde uplná je definována jako množina všech kombinací hodnot

proměnných se symboly 0, 1. Těchto kombinací je 2^n a představují n-tici o délce n. Obor hodnot má délku 1, proto výstup booleovské funkce je roven 0 nebo 1. Neúplná booleovská funkce může mít na výstupu tři hodnoty: 0, 1 a X. Hodnota X znamená, že na tomto výstupu nezáleží. Tato hodnota se přiřazuje výstupu hlavně v situacích, kdy vstupní kombinace nemůže vůbec nastat.[1]

2.4 Pravdivostní tabulky

K určení jednoho výstupního stavu může být rozhodující logický stav dvou nebo více vstupních proměnných. Tento vzájemný vztah přehledně popisuje *pravdivostní tabulka*. Vstupy označujeme A, B, výstupním symbolem je Y.[2]

2.5 Logické operace

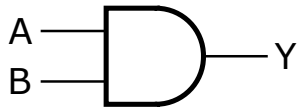
K vyjádření libovolné logické funkce se v Booleově algebře používají jen tři základní funkce: logický součet, logický součin a logická negace. Těmito základními funkcemi můžeme vyjádřit libovolnou logickou operaci. Logické proměnné a logické funkce i konstanty nabývají v Booleové algebře jen dvou hodnot, takže vlastně tato algebra počítá ve dvojkové soustavě.[2] Logické operace se znázorňují pomocí značek dle standardu IEEE Std 91a-1991.[5]

2.5.1 Logický součin

Logický součin je logická funkce ve tvaru:

$$Y = A \cdot B$$

Kde Y s indexem značí výstup a symboly A a B jsou vstupními proměnnými a symbol $\cdot$ představuje operátor logického součinu. V některých literaturach je označován jako konjunkce, operace AND, logické násobení nebo logický součin. Logický člen, který realizujeme funkcí logického součinu, se označuje AND nebo v českém vyjádření čteme A.[1, 2]



Obrázek 2.2: Značka hradla AND

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

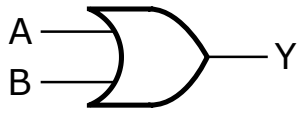
Tabulka 2.1: Pravdivostní tabulka pro AND

2.5.2 Logický součet

Logický součet je logická funkce ve tvaru:

$$Y = A + B$$

Kde symbol + představuje operátor logického součtu. Logický člen, který realizuje funkci logického součinu, se označuje OR v českém vyjádření čteme NEBO.[2]



Obrázek 2.3: Značka hradla OR

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Tabulka 2.2: Pravdivostní tabulka pro OR

2.5.3 Logická negace

Nejjednodušší funkcí je logická negace. V algebraickém tvaru se vyjadřuje:

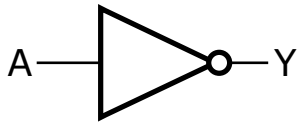
$$Y = \overline{A}$$

V některé literatuře se můžeme setkat s pojmem "A non". Hradlo se označuje NOT. Toto hradlo jako jediné provádí unární operaci, proto má jen jednu vstupní proměnnou. [2]

2.5.4 Negovaný logický součin

Negací logického součinu je logická funkce ve tvaru:

$$Y = \overline{A \cdot B}$$

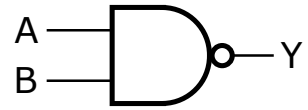


Obrázek 2.4: Značka hradla NOT

A	Y
0	1
1	0

Tabulka 2.3: Pravdivostní tabulka pro NOT

Tato funkce vzniká negací funkce AND, kde nadtržení znamená obrácení hodnoty výstupní proměnné. Na obrázku 2.5 je negace znázorněna kroužkem na výstupu hradla. Logický člen, který realizuje negovaný logický součin se označuje NAND. Alternativní názvy pro toto hradlo jsou negace konjunkce, operace NAND, negace AND, negace logického násobení, negovaný logický součin nebo Shefferova funkce. [1, 2]



Obrázek 2.5: Značka hradla NAND

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Tabulka 2.4: Pravdivostní tabulka pro NAND

2.5.5 Negovaný logický součet

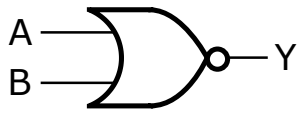
Negací logického součtu je logická funkce ve tvaru:

$$Y = \overline{A + B}$$

Případně ekvivalentně zápisem:

$$Y = \overline{A} * \overline{B}$$

Tato funkce vzniká negací OR. Logický člen, který realizuje negovaný logický součet se označuje NOR. Alternativní názvy pro toto hradlo jsou negace disjunkce, operace NOR, negace OR, negace logického součtu nebo Peirceova funkce.[1, 2]



Obrázek 2.6: Značka hradla NOR

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

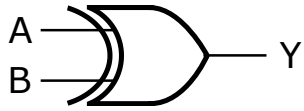
Tabulka 2.5: Pravdivostní tabulka pro NOR

2.5.6 Exkluzivní logický součet

Takzvaný exkluzivní OR tedy XOR. Je poslední běžně využívané hradlo. Funkci to lze znázornit jako:

$$Y = \overline{A} * B + A * \overline{B}$$

Odpovídá matematickému součtu modulo 2. Alternativní názvy pro hradlo XOR je non-equivalence, exclusive OR.[1]



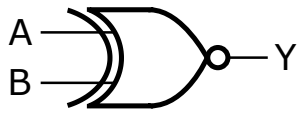
Obrázek 2.7: Značka hradla XOR

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Tabulka 2.6: Pravdivostní tabulka pro XOR

2.5.7 Exkluzivní negovaný logický součet

Vychází z hradla XOR, ale výstup je negovaný. Alternativní názvy jsou ekvivalence, exclusive NOR. [1]



Obrázek 2.8: Značka hradla XNOR

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

Tabulka 2.7: Pravdivostní tabulka pro XNOR

2.6 Bitové operace

Je třeba si uvědomit rozdíl mezi bitovými a binárními operacemi. Oba typy mohou být realizovány pomocí soustavy logických operací. Výstupem bitové operace je skupina bitů pozměněná bitovou operací. Implementace těchto operací je již v základních instrukčních sadách procesorů. O kterých je více zmíněno ve třetí kapitole. Na rozdíl od toho binární operace je totožná s aritmetickou pouze její realizace je ve dvojkové soustavě.[3, 8]

2.6.1 Bitový součin

Nejzákladnější bitová operace je bitový součin. Máme-li čísla A, B a chceme provést bitový součin můžeme zapsat pro každý bit:

$$Y = A + B$$

Bitový součin se řídí podle tabulky 2.1 logické funkce AND pro každý bit. Pro binární součin, tak jako u aritmetické operace, je potřeba brát v úvahu přenos do vyššího řádu a z tohoto důvodu může být vektor součtu větší o jeden řád. U bitového součinu přenos do vyššího řádu neprobíhá.[6, 7]

2.6.2 Bitový součet

Druhá bitová operace je bitový součet. Pro každý bit platí:

$$Y = A \cdot B$$

Bitový součin se řídí podle tabulky 2.2 logické funkce OR.

2.6.3 Bitová negace

Unární operace, provádí logickou negaci každého bitu. Má pouze jeden operand. V případě nuly bude jedna a naopak. U bitové negace není důležité, zda je datový typ znaménkový nebo bezznaménkový. Při negaci bitů se nebere ohled na znaménko čísla, nejde tedy o ekvivalentní operaci se změnou znaménka reprezentovaného čísla.[3, 6]

```
#include <cstdint>

int main() {
  int8_t cislo = 7; // 0x07, 7
  int8_t bitova_negace = ~cislo; // 0xF8, -8
  uint8_t bitova_negace_bez_znamenka = ~cislo; // 0xF8, 248
  int8_t znamenkova_negace = -cislo; // 0xF9, -7
}
```

Výpis 1: Ukázka negace v programovacím jazyce C

2.6.4 Bitový posun

Bitový posun rozlišujeme na posun vlevo nebo posun vpravo o jeden či více bitů. Hodnoty jednotlivých bitů se posunou a chybějící bity jsou doplněny nulami. V připadě bitového posunu doprava může být místo nul doplněna hodnota bitu nejvíce vlevo.[9]

```
#include <cstdint>

int main() {
  int8_t cislo = 7; //00000111, 0x07, 7
  int8_t cislo_posun_doprava = cislo >> 1; //00000011, 0x03, 3
  int8_t cislo_posun_doleva = cislo << 1; //00001110, 0x0E, 14
}
```

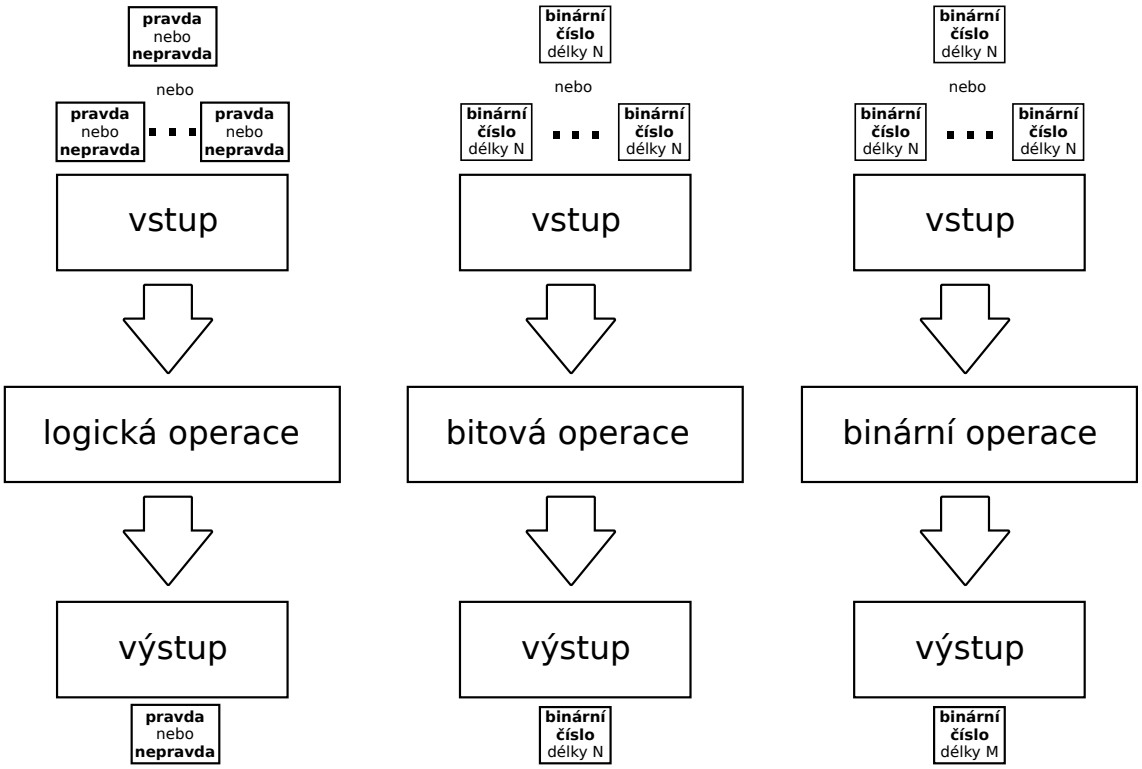
Výpis 2: Ukázka bitového posunu v programovacím jazyce C

2.6.5 Bitová rotace

Bitové rotace se podobají bitovým posunům. Rozdíl je v tom, že bit který je při posunu vysouván z hodnoty ven, je při rotaci opět vkládán na opačném konci. Tato operace nebývá dostupná ve vyšších programovacích jazycích, je však ve instrukční sadě mnoha procesorů.[9, 12]

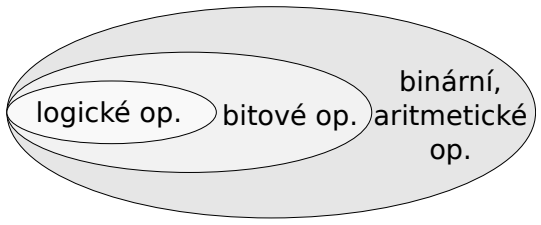
2.7 Rozdíl mezi logickými a bitovými operacemi

Hradla vykonávají logické operace pro jeden nebo více jednobitových vstupů. Výstup nabývá pouze logických proměnných. Bitové operace umí provádět úlohu pro skupinu bitů o stejné velikosti N, jak je uvedeno na obrázku 2.9. Výsledkem bitové operace je nová skupina bitů olivněná operací o shodné velikosti N bitů. Pro lepší přiblížení je doplněno, jak je to u aritmetických/binárních operací. Rozdílem oproti bitovým opracím je, že výsledná velikost bitu může nabývat hodnot $M \geq N$ při přenosu do vyššího řádu.[3, 6, 7]



Obrázek 2.9: Rozdíl logických, bitových a binárních operací

Na obrázku 2.10 je znázorněna návaznost mezi operacemi.



Obrázek 2.10: Návaznost operací

3 Praktické použití

V dnešní době si svět bez počítačů už snad ani nedokážeme představit. Vycházejí z digitálních obvodů, které používá většina dnešní elektroniky. Výrábějí se od těch nejjednodušších, jako jsou logická hradla, čítače, paměti a AD/DA převodníky, až po složitější mikrokontroléry nebo programovatelná hradlová pole. Logické a bitové operace jsou již implementovány v některých instrukčních sadách procesorů, pro ukázku jsem zvolil procesor i486.

3.1 Logické a bitové instrukce procesoru i486

Z celého instrukčního souboru procesoru i486 a jeho následníků se využívá v běžné praxi ani ne polovina všech instrukcí celočíselné jednotky ALU. Procesory i486 obsahují 8 základních registrů velikosti 32 bitů pro všeobecné použití. Dále 6 registrů segmentových, stavový registr a čítač instrukcí. Instrukce ovlivňují obsah příznakového registru procesoru. Logické instrukce mění SF a ZF, bitové posuny i CF. Stavový registr FLAGS obsahuje stavové bity. Pro potřeby této práce jsou podstatné: ZF (zero flag), CF (carry), SF (signum).

Mezi logické a bitové instrukce patří :

- *AND cíl, zdroj* Instrukce provede bitově cíl = cíl and zdroj.
- *TEST cíl, zdroj* Instrukce provede bitově cíl and zdroj. Jde o operaci AND bez uložení výsledku.
- *OR cíl, zdroj* Instrukce provede bitově cíl = cíl or zdroj.
- *XOR cíl, zdroj* Instrukce provede bitově cíl = cíl xor zdroj.
- *NOT cíl* Instrukce provede negaci všech bitů operandu.
- *SHL/SAL cíl, kolik* Bitový i aritmetický posun doleva operandu cíl o požadovaný počet bitů. Operand kolik je konstanta nebo registr CL.
- *SHR cíl, kolik* Bitový posun doprava operandu cíl o požadovaný počet bitů. Operand kolik je konstanta nebo registr CL.
- *SAR cíl, kolik* Aritmetický posun doprava operandu cíl o požadovaný počet bitů. Operand kolik je konstanta nebo registr CL.
- *ROL cíl, kolik* Bitová rotace doleva operandu cíl o požadovaný počet bitů. Operand kolik je konstanta nebo registr CL.
- *ROR cíl, kolik* Bitová rotace doprava operandu cíl o požadovaný počet bitů. Operand kolik je konstanta nebo registr CL.
- *RCL cíl, kolik* Bitová rotace doleva přes CF operandu cíl o požadovaný počet bitů. Operand kolik je konstanta nebo registr CL.

- *RCR cíl, kolik* Bitová rotace doprava přes CF operandu cíl o požadovaný počet bitů.Operand kolik je konstanta nebo registr CL.
- *BT cíl, číslo* Zkopíruje do CF hodnotu bitu daného operandem číslo z operanducíl.
- *BTR cíl, číslo* Zkopíruje do CF hodnotu bitu daného operandem číslo z operandu cíl a nastaví jej na nulu.
- *BTS cíl, číslo* Zkopíruje do CF hodnotu bitu daného operandem číslo z operandu cíl a nastaví jej na jedničku.
- *BTC cíl, číslo* Zkopíruje do CF hodnotu bitu daného operandem číslo z operandu cíl a provede jeho negaci.
- *SETcc cíl* Nastaví cíl na hodnotu 0/1 podle toho, zda je splněna požadovaná podmínka (podmínky viz podmíněné skoky).
- *SHRD/SHLD cíl, zdroj, kolik* Provede nasunutí kolik bitů ze zdroje do cíle. Zdroj se nemění.

[8]

3.2 Reprezentace v programovacích jazycích

Logické operace mohou nabývat pouze dvou hodnot a v programování mají proto vyhrazen svůj vlastní datový typ $Boolean$, nabývá pouze hodnot: $True$ nebo $False$. V jazyce C++ se tento typ označuje $bool$, v Javě $boolean$ apod.

Pro bitové operace se používá datový typ pro přirozená nebo celá čísla $Integer$. V jazycích C++ a Java se označuje int. V C++ se rozděluje na znaménkový $signed$, který může nabývat i záporné hodnoty, a bezznaménkový $unsigned$. Ve výchozím stavu je $signed$, pokud není uvedeno klíčové slovo $unsigned$.

Jelikož jsou v programovacích jazycích rozlišeny operátory pro logické a bitové operace, je možné použít operátory logických operací i nad datovým typem int. Podrobné vysvětlení chování je znázorněno na praktických příkladech v následujícím textu.

Následující podkapitola zobrazuje použití logických a bitových operací přímo ve zdrojovém kódu v jednotlivých jazycích. Výpis 3 ukazuje logické operace v programovacím jazyce C++. Výpisy 4 a 5 obsahují obdobnou úlohu v jazycích Java a Pascal. Výstup spuštěného programu je vidět na výpisech 6, 7 a 8. Seznam všech základních logickách operátorů poskytovaných jednotlivými jazyky je zachycen v tabulce 3.1.[9, 10, 11]

```
#include <iostream>

bool a = false;
bool b = true;
int c = 5;
int d = 7;

int main() {
```

```
  std::cout << "Logický součin pro bool . . . . . " ;
  std::cout << a << " AND " << b << " = " << (a && b) << std::endl;

  std::cout << "Logický součet pro bool . . . . . " ;
  std::cout << a << " OR " << b << " = " << (a || b) << std::endl;

  std::cout << "Logická negace pro bool . . . . . " ;
  std::cout << "NOT " << a << " = " << !a << std::endl;

  std::cout << "Negovaný logický součin pro bool  . . . . . " ;
  std::cout << a << " NAND " << b << " = " << !(a && b) << std::endl;

  std::cout << "Negovaný logický součet pro bool  . . . . . " ;
  std::cout << a << " NOR " << b << " = " << !(a || b) << std::endl;

  std::cout << "Exkluzivní logický součet pro bool  . . . . . " ;
  std::cout << a << " XOR " << b << " = " << (a ^ b) << std::endl;

  std::cout << "Negovaný exkluzivní logický součet pro bool . " ;
  std::cout << a << " XNOR " << b << " = " << !(a ^ b) << std::endl;

  std::cout << std::endl;

  std::cout << "Logický součin pro int  . . . . . " ;
  std::cout << c << " AND " << d << " = " << (c && d) << std::endl;

  std::cout << "Logický součet pro int  . . . . . " ;
  std::cout << c << " OR " << d << " = " << (c || d) << std::endl;

  std::cout << "Logická negace pro int  . . . . . " ;
  std::cout << "NOT " << c << " = " << !c << std::endl;
}
```

Výpis 3: Ukázka použití logických operátorů v programovacím jazyku C++

```
public class LogickéOperace {
  public static void main(String[] args) {
    boolean a = false;
    boolean b = true;
    int c = 5;
    int d = 7;

    System.out.print("Logický součin pro bool . . . . . ");
    System.out.println(a + " AND " + b + " = " + (a && b));

    System.out.print("Logický součet pro bool . . . . . ");
    System.out.println(a + " OR " + b + " = " + (a || b));

    System.out.print("Logická negace pro bool . . . . . ");
    System.out.println("NOT " + a + " = " + !a);

    System.out.print("Negovaný logický součin pro bool  . . . . . ");
    System.out.println(a + " NAND " + b + " = " + !(a && b));
```

```
        System.out.print("Negovaný logický součet pro bool  . . . . . ");
        System.out.println(a + " NOR " + b + " = " + !(a || b));

        System.out.print("Exkluzivní logický součet pro bool  . . . . . ");
        System.out.println(a + " XOR " + b + " = " + (a ^ b));

        System.out.print("Negovaný exkluzivní logický součet pro bool . ");
        System.out.println(a + " XNOR " + b + " = " + !(a ^ b));

        System.out.println();

        System.out.print("Logický součin pro int  . . . . . ");
        System.out.println(c + " AND " + d + " = " + ((c != 0) && (d != 0)));

        System.out.print("Logický součet pro int  . . . . . ");
        System.out.println(c + " OR " + d + " = " + ((c != 0) || (d != 0)));

        System.out.print("Logická negace pro int  . . . . . ");
        System.out.println("NOT " + c + " = " + !(c != 0));
    }
}
```

Výpis 4: Ukázka použití logických operátorů v programovacím jazyku Java

```
program LogickeOperace;
var
   a: boolean = false;
   b: boolean = true;
   c: integer = 5;
   d: integer = 7;
begin
   write('Logický součin pro bool . . . . . ');
   writeln(a, ' AND ', b, ' = ', (a and b));

   write('Logický součet pro bool . . . . . ');
   writeln(a, ' OR ', b, ' = ', (a or b));

   write('Logická negace pro bool . . . . . ');
   writeln('NOT ', a, ' = ', not a);

   write('Negovaný logický součin pro bool  . . . . . ');
   writeln(a, ' NAND ', b, ' = ', not (a and b));

   write('Negovaný logický součet pro bool  . . . . . ');
   writeln(a, ' NOR ', b, ' = ', not (a or b));

   write('Exkluzivní logický součet pro bool  . . . . . ');
   writeln(a, ' XOR ', b, ' = ', (a xor b));

   write('Negovaný exkluzivní logický součet pro bool . ');
   writeln(a, ' XNOR ', b, ' = ', not (a xor b));

   writeln();
```

```
    write('Logický součin pro int  . . . . . . . . . . . . . ');
    writeln(c, ' AND ', d, ' = ', ((c <> 0) and (d <> 0)));

    write('Logický součet pro int  . . . . . . . . . . . . . ');
    writeln(c, ' OR ', d, ' = ', ((c <> 0) or (d <> 0)));

    write('Logická negace pro int  . . . . . . . . . . . . . ');
    writeln('NOT ', c, ' = ', not (c <> 0));
end.
```

Výpis 5: Ukázka použití logických operátorů v programovacím jazyku Pascal

```
Logický součin pro bool . . . . . . . . . . . . 0 AND 1 = 0
Logický součet pro bool . . . . . . . . . . . . 0 OR 1 = 1
Logická negace pro bool . . . . . . . . . . . . NOT 0 = 1
Negovaný logický součin pro bool . . . . . . . 0 NAND 1 = 1
Negovaný logický součet pro bool . . . . . . . 0 NOR 1 = 0
Exkluzivní logický součet pro bool . . . . . . 0 XOR 1 = 1
Negovaný exkluzivní logický součet pro bool . 0 XNOR 1 = 0

Logický součin pro int . . . . . . . . . . . . 5 AND 7 = 1
Logický součet pro int . . . . . . . . . . . . 5 OR 7 = 1
Logická negace pro int . . . . . . . . . . . . NOT 5 = 0
```

Výpis 6: Výstup pro výpis 3

```
Logický součin pro bool . . . . . . . . . . . . false AND true = false
Logický součet pro bool . . . . . . . . . . . . false OR true = true
Logická negace pro bool . . . . . . . . . . . . NOT false = true
Negovaný logický součin pro bool . . . . . . . false NAND true = true
Negovaný logický součet pro bool . . . . . . . false NOR true = false
Exkluzivní logický součet pro bool . . . . . . false XOR true = true
Negovaný exkluzivní logický součet pro bool . false XNOR true = false

Logický součin pro int . . . . . . . . . . . . 5 AND 7 = true
Logický součet pro int . . . . . . . . . . . . 5 OR 7 = true
Logická negace pro int . . . . . . . . . . . . NOT 5 = false
```

Výpis 7: Výstup pro výpis 4

```
Logický součin pro bool . . . . . . . . . . . . FALSE AND TRUE = FALSE
Logický součet pro bool . . . . . . . . . . . . FALSE OR TRUE = TRUE
Logická negace pro bool . . . . . . . . . . . . NOT FALSE = TRUE
Negovaný logický součin pro bool . . . . . . . FALSE NAND TRUE = TRUE
Negovaný logický součet pro bool . . . . . . . FALSE NOR TRUE = FALSE
Exkluzivní logický součet pro bool . . . . . . FALSE XOR TRUE = TRUE
Negovaný exkluzivní logický součet pro bool . FALSE XNOR TRUE = FALSE

Logický součin pro int . . . . . . . . . . . . 5 AND 7 = TRUE
Logický součet pro int . . . . . . . . . . . . 5 OR 7 = TRUE
Logická negace pro int . . . . . . . . . . . . NOT 5 = FALSE
```

Výpis 8: Výstup pro výpis 5

	C/C++	Java	Pascal
AND	$\&\&, and$	$\&\&$	and
OR	$\|\|, or$	$\|\|$	or
NOT	$!, not$	!	not

Tabulka 3.1: Tabulka reprezentující logické operátory

Vedle logických operací je možné v programovacích jazycích zpracovávat i operace bitové. Výpisy 9, 10 a 11 ukazují zdrojový kód v jazycích C++, Java a Pascal, ve kterém jsou nad celočíselnými proměnnými aplikovány operace bitového součinu, součtu, exkluzivního bitového součtu, negace a posunu v obou směrech. Výstup programu je zachycen na výpisu 12. Základní bitové operátory zachycuje tabulka 3.2.

```
#include <climits>
#include <iostream>

const int INT_BITS = sizeof(int)*CHAR_BIT;

int a = 5;
int b = 7;

int main() {
  std::cout << "Bitový součin . . . . . ";
  std::cout << a << " AND " << b << " = " << (a & b) << std::endl;

  std::cout << "Bitový součet . . . . . ";
  std::cout << a << " OR " << b << " = " << (a | b) << std::endl;

  std::cout << "Bitová negace . . . . . ";
  std::cout << "NOT " << a << " = " << ~a << std::endl;

  std::cout << "Bitový posun doleva . . . . . ";
  std::cout << a << " SHL " << b << " = " << (a << b) << std::endl;

  std::cout << "Bitový posun doprava  . . . . . ";
  std::cout << a << " SHR " << b << " = " << (a >> b) << std::endl;

  std::cout << "Bitová rotace doleva  . . . . . ";
  std::cout << a << " ROTL " << b << " = " << ((a << b) | (a >> INT_BITS-b
      )) << std::endl;

  std::cout << "Bitová rotace doprava . . . . . ";
  std::cout << a << " ROTR " << b << " = " << ((a >> b) | (a << INT_BITS-b
      )) << std::endl;
}
```

Výpis 9: Ukázka použití bitových operátorů v programovacím jazyku C++

```
public class BitovéOperace {
  public static void main(String[] args) {
    int a = 5;
```

```
        int b = 7;

        System.out.print("Bitový součin . . . . . ");
        System.out.println(a + " AND " + b + " = " + (a & b));

        System.out.print("Bitový součet . . . . . ");
        System.out.println(a + " OR " + b + " = " + (a | b));

        System.out.print("Bitová negace . . . . . ");
        System.out.println("NOT " + a + " = " + ~a);

        System.out.print("Bitový posun doleva . . . . . ");
        System.out.println(a + " SHL " + b + " = " + (a << b));

        System.out.print("Bitový posun doprava  . . . . . ");
        System.out.println(a + " SHR " + b + " = " + (a >> b));

        System.out.print("Bitová rotace doleva  . . . . . ");
        System.out.println(a + " ROTL " + b + " = " + ((a << b) | (a >>
            Integer.SIZE-b)));

        System.out.print("Bitová rotace doprava . . . . . ");
        System.out.println(a + " ROTR " + b + " = " + ((a >> b) | (a <<
            Integer.SIZE-b)));
    }
}
```

Výpis 10: Ukázka použití bitových operátorů v programovacím jazyku Java

```
program BitoveOperace;
const
   INT_BITS = sizeof(integer)*8;
var
   a: integer = 5;
   b: integer = 7;
begin
   write('Bitový součin . . . . . ');
   writeln(a, ' AND ', b, ' = ', (a and b));

   write('Bitový součet . . . . . ');
   writeln(a, ' OR ', b, ' = ', (a and b));

   write('Bitová negace . . . . . ');
   writeln('NOT ', a, ' = ', not a);

   write('Bitový posun doleva . . . . . ');
   writeln(a, ' SHL ', b, ' = ', (a shl b));

   write('Bitový posun doprava  . . . . . ');
   writeln(a, ' SHR ', b, ' = ', (a shr b));

   write('Bitová rotace doleva  . . . . . ');
   writeln(a, ' ROTL ', b, ' = ', ((a shl b) or (a shr (INT_BITS-b))));
```

```
    write('Bitová rotace doprava . . . . . ');
    writeln(a, ' ROTR ', b, ' = ', ((a shr b) or (a shl (INT_BITS-b))));
end.
```

Výpis 11: Ukázka použití bitových operátorů v programovacím jazyku Pascal

```
Bitový součin . . . . . 5 AND 7 = 5
Bitový součet . . . . . 5 OR 7 = 7
Bitová negace . . . . . NOT 5 = -6
Bitový posun doleva . . . . . 5 SHL 7 = 640
Bitový posun doprava . . . . . 5 SHR 7 = 0
Bitová rotace doleva . . . . . 5 ROTL 7 = 640
Bitová rotace doprava . . . . . 5 ROTR 7 = 167772160
```

Výpis 12: Výstup pro výpis 9, 10 a 11

	C/C++	Java	Pascal
AND	$\&, bitand$	$\&$	$\&, and$
OR	$\|, bitor$	$\|$	$\|, or$
NOT	$\sim, compl$	$\sim$	$\sim, not$
XOR	$\wedge, xor$	$\wedge$	$\wedge, xor$
bitový posun vlevo	$<<$	$<<$	$<<, shl$
bitový posun vpravo	$>>$	$>>$	$>>, shr$
bitový posun vpravo bez znaménka		$>>>$	

Tabulka 3.2: Tabulka reprezentující bitové operátory

3.3 Hardwarová reprezentace

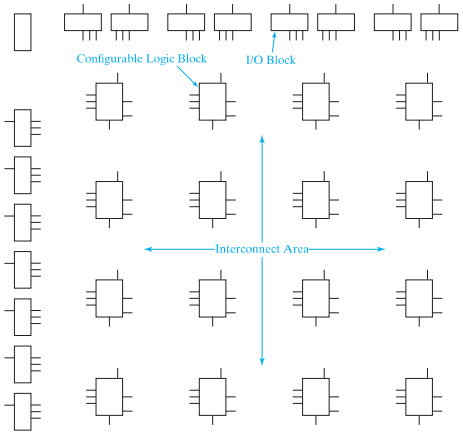
Následující části kapitoly jsou zaměřeny na praktickou část, zpracovanou na FPGA kitu a poté hradlová stavebnice.

3.3.1 FPGA obecně

FPGA je programovatelné hradlové pole. Skládá se z integrovaných obvodů, které obsahují celou řadu shodné logiky. Jsou uživatelsky konfigurovatelné pro každou logickou buňku. Skládají se z konfigurovatelných logických bloků viz obrázek 3.1.[3]

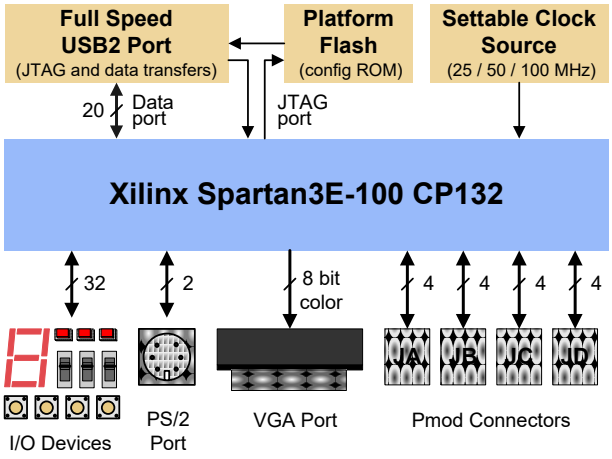
3.3.2 Seznámení s FPGA kitem Digilent Basys2

Digilent Basys2 se skládá z 100 000 CLBs řady Xilinx Spartan 3E FPGA. Pro komunikaci s PC je použit řadič Atmel AT90USB2, který Basys2 poskytuje napájení a programovací, datové rozhraní. Dále na kitu nalezneme 8 LED diod, 4 číselný digitální 7-mi segmentový display, 8 přepínatelných tlačítek a 4 normální tlačítka. Uživatelsky nastavitelný hodinový signál (25/50/100MHz) se soketem pro druhý hodinový signál. Pro naši potřebu nebude použit, ale taky se na stavebnici nachází PS/2 port a 8-bitový VGA Port. Poslední věcí,



Obrázek 3.1: Obecné schéma FPGA obvodu [13]

kterou tento kit disponuje, jsou čtyři 6-pinové Pmod konektory, které mohou kit rozšířit o další součásti, například další VGA port. Všechny vstupně-výstupní porty jsou opatřeny ESD diodovou ochranou obvodu, která má za cíl ochránit obvody před nežádoucím elektrostatickým nábojem. Ten by mohl obvod poškodit, případně i zničit.[13]

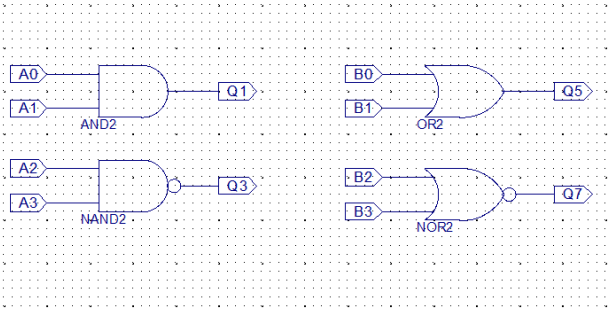


Obrázek 3.2: Blokové schéma Basys 2 [13]

Vývojové prostředí pro realizaci kódu od společnosti Xilinx je ISE WebPACK a program pro komunikaci s FPGA kitem je Adept Systems. Aby výstupní soubor správně fungoval je třeba vývojové prostředí nastavit na správné programovatelné hradlové pole, model a jeho řadu. V našem případě se jedná o Xilinx Spartan 3E-100 CP132.

3.3.3 Implementace na Digilent Basys2

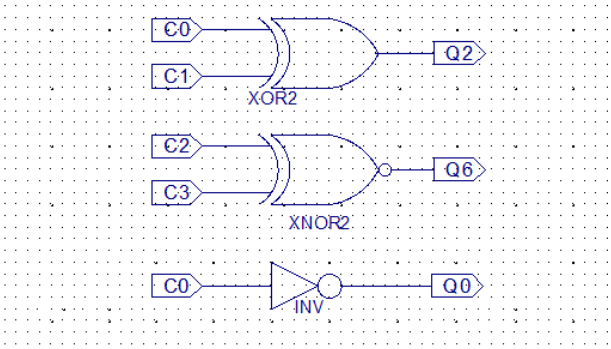
Reprezentoval jsem základní logická hradla schématicky a následně připojil k odpovídajícím vstupům na kitu, které tvoří přepínače, tlačítka a výstup jsou LED diody. Pro případ logické jedničky dioda svítí, pokud nesvítí - reprezentuje logickou nulu.



Obrázek 3.3: Blokové schéma hradel AND, NAND ,OR a NOR v Xilinxu

Obrázky 3.3 a 3.4 schématicky demonstrují realizace VHDL na výpisu 13 Vstupy u obrázku 3.3 $A_0 - A_4$ jsou propojeny s přepínači $SW_7 - SW_4$ a vstupy $B_0 - B_4$ jsou propojeny s přepínači $SW_3 - SW_0$. Výstupy hradel Q_1, Q_3, Q_5, Q_7 pak s diodami LD_6, LD_4, LD_2, LD_0.

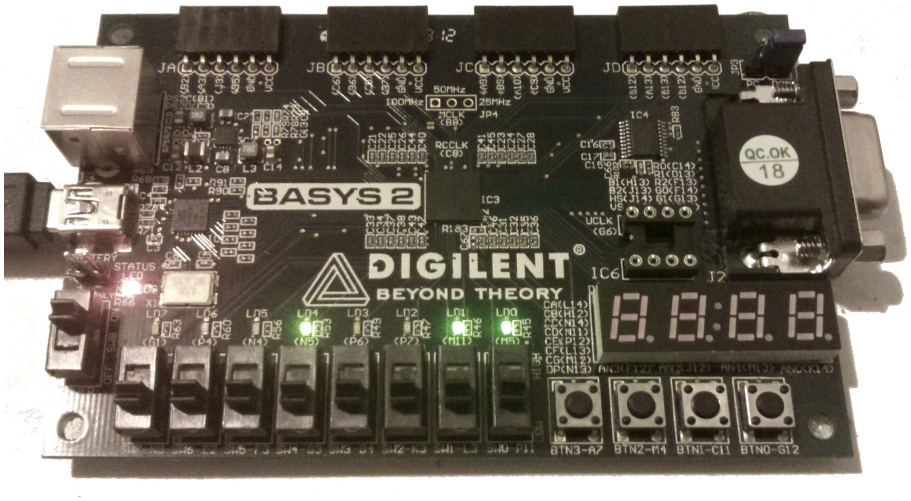
Vstupy u obrázku 3.4 $C_0 - C_3$ jsou propojeny s tlačítky $BTN_3 - BTN_0$. Invertory jsou dva, aby při zmáčknutí tlačítka C_0 dioda na pozici Q_0 svítila.



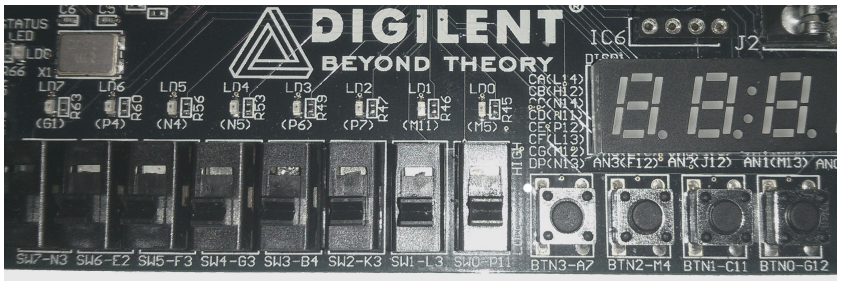
Obrázek 3.4: Blokové schéma hradel XOR, XNOR a dvojitý invertor v Xilinxu

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity gates is
  port (
    A0, A1, A2, A3, B0, B1, B2, B3, C0, C1, C2, C3: in std_logic;
    Q0, Q1, Q2, Q3, Q5, Q6, Q7: out std_logic
  );
end entity gates;

architecture main of gates is
begin
Q0 <= not C0; Q1 <= A0 and A1; Q2 <= C0 xor C1;
Q3 <= A2 nand A3; Q5 <= B0 or B1; Q6 <= C2 xnor C3;  Q7 <= B2 nor B3;
end architecture;
}
```

Výpis 13: Realizace logických hradel pomocí VHDL



Obrázek 3.5: Realizace logických operací



Obrázek 3.6: Značení součástek na FPGA kitu

```
NET A0 LOC = "N3"; NET A1 LOC = "E2"; NET A2 LOC = "F3"; NET A3 LOC = "G3"
    ;
NET B0 LOC = "B4"; NET B1 LOC = "K3"; NET B2 LOC = "L3"; NET B3 LOC = "P11
    ";
NET C0 LOC = "A7"; NET C1 LOC = "M4"; NET C2 LOC = "C11"; NET C3 LOC = "
    G12";
NET Q0 LOC = "G1"; NET Q1 LOC = "P4"; NET Q2 LOC = "N4"; NET Q3 LOC = "N5"
    ;
NET Q4 LOC = "P6"; NET Q5 LOC = "P7"; NET Q6 LOC = "M11"; NET Q7 LOC = "M5
    ";
```

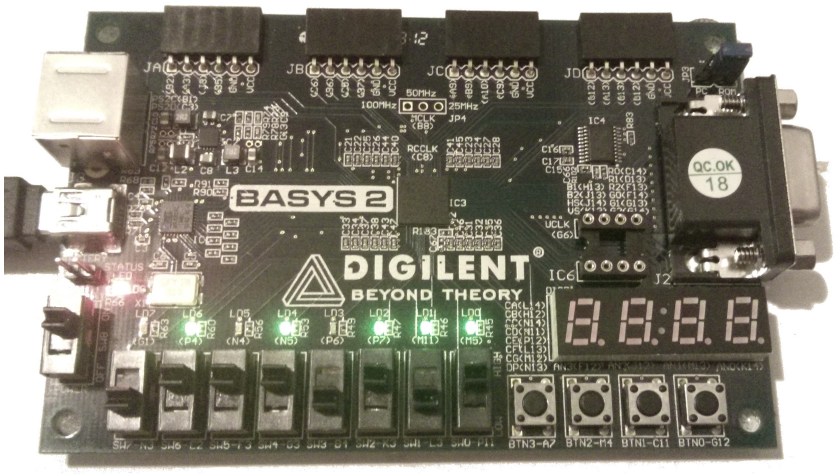
Výpis 14: Připojení proměnných v kódu ke kitu a součástkám na desce.

Výpis 13 demonstruje použití logických hradel pomocí VHDL kódu.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity bitwiseAND_OR is
  port (
```

```
    A0, A1, A2, A3, B0, B1, B2, B3: in std_logic;
    Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7: out std_logic
  );
end entity bitwiseAND_OR;

architecture main of bitwiseAND_OR is
begin
Q0 <= A0 and B0; Q1 <= A1 and B1; Q2 <= A2 and B2; Q3 <= A3 and B3;
Q4 <= A0 or B0; Q5 <= A1 or B1; Q6 <= A2 or B2; Q7 <= A3 or B3;
end architecture;
}
```

Výpis 15: Ukázka použití hradel pro bitové operace AND a OR



Obrázek 3.7: Realizace bitových operací AND, OR

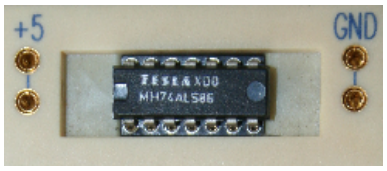
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity bitwiseNOT_XOR is
  port (
    A0, A1, A2, A3, B0, B1, B2, B3: in std_logic;
    Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7: out std_logic
  );
end entity bitwiseNOT_XOR;

architecture main of bitwiseNOT_XOR is
begin
Q0 <= (not A0); Q1 <= (not A1); Q2 <= (not A2); Q3 <= (not A3);
Q4  <= A0 xor B0; Q5 <= A1 xor B1; Q6 <= A2 xor B2; Q7 <= A3 xor B3;
end architecture;
}
```

Výpis 16: Ukázka použití hradel pro bitové operace NOT a XOR

3.4 Reprezentace pomocí stavebnice Modular RC 2000

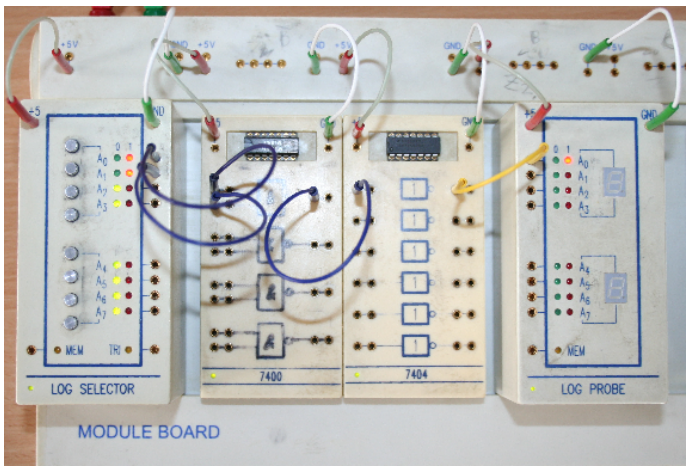
Provedl jsem demonstraci základních hradel na stavebnici Modular RC2000, která v našem případě používá pro vstup Log Selector a pro výstup Log Probes. Každý modul musí být nezávisle napájen ze zdroje.

Každý hradlový modul obsahuje typ integrovaného obvodu. Na obrázku 3.8 je ukázka integrovaného obvodu se čtyřmi dvouvstupými XOR hradly.



Obrázek 3.8: Integrovaný obvod se čtymi dvouvstupými XOR hradly

Hradlo AND jsem vytvořil pomoci dvouvstupého NANDu a jednovstupého invertoru. Na následujících obrázcích je vždy vyobrazen stav, kdy na vstupu jsou dvě jedničky.

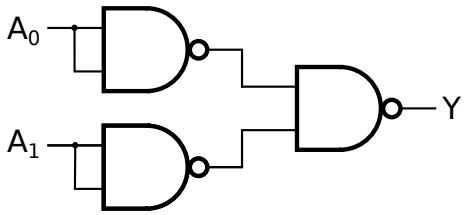


Obrázek 3.9: Zapojení hradla AND

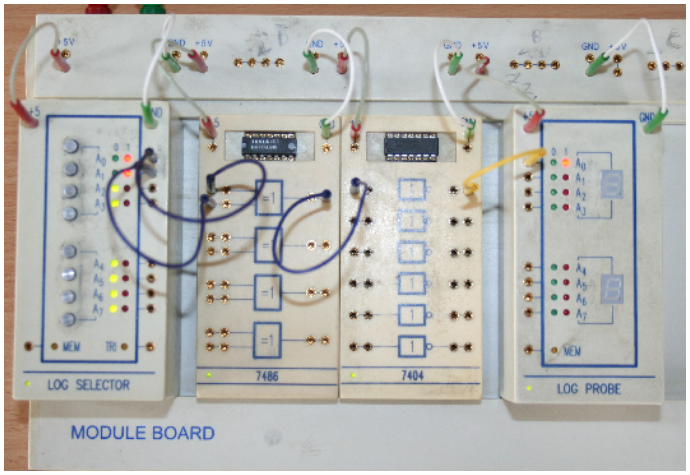
Pro hradlo OR jsem s pomocí tří NAND hradel vytvořil OR. Viz schéma realizovaného zapojení na obrázku 3.10.

Hradlo NAND má nejjednodušší zapojení viz obrázek 3.10. Bylo již zapojeno schématicky a teď realizace pomocí modulů.

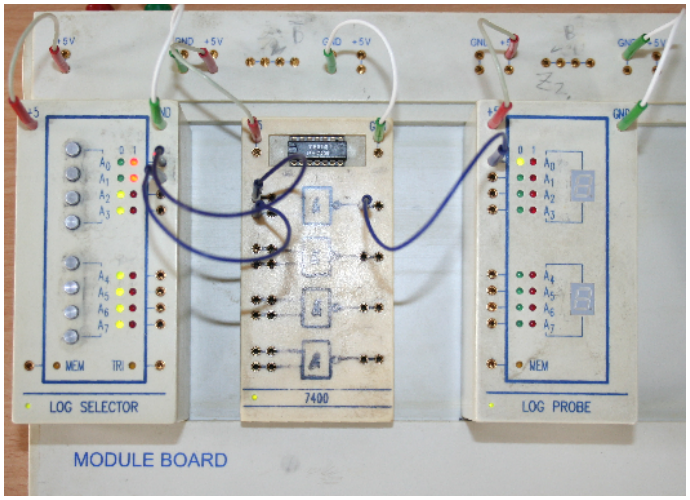
Bitové operace za pomocí logických hradel jsou znázorněny schématicky no obrázku 3.13 a 3.14 .



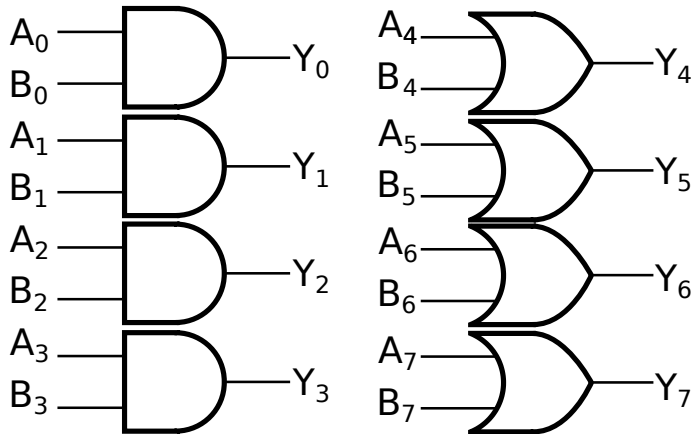
Obrázek 3.10: Schéma zapojení hradla OR pomocí NAND hradel



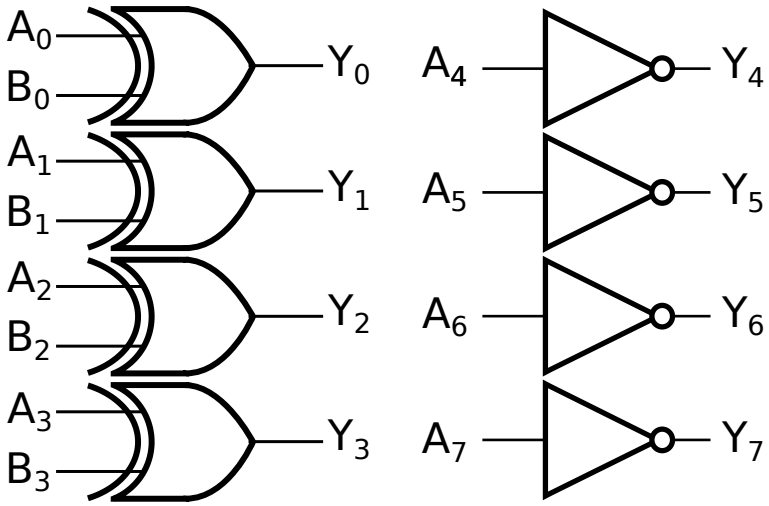
Obrázek 3.11: Zapojení hradla XNOR



Obrázek 3.12: Zapojení hradla NAND



Obrázek 3.13: Schéma zapojení půl Bajtového binárního součinu a součtu



Obrázek 3.14: Schéma zapojení půl Bajtového exkluzivního binárního součtu a negace

4 Interaktivní animace

Interaktivní animace vznikla za účelem lepšího pochopení výše zmiňovaných operací. Cílem bylo přiblížit problematiku a hlavně ukázat rozdíl a vnitřní funkčnost číslicových obvodů. Animace bude připravena na webovou stránku do LMS. Dále bude dostupný manuál, který bude obsahovat popis a teorii k animacím.

4.1 Návrh

Program, který vznikal v průběhu této práce, má za úkol demonstrovat základní logické hradla a bitové operace podle požadavků mého vedoucího bakalářské práce. Výsledný program měl být snadno integrovatelný do libovolného webového výukového systému. Na výběr tedy bylo z několika možných technologií: Adobe Flash, Microsoft Silverlight, HTML5 a případně i další.

Protože technologie Adobe Flash a Microsoft Silverlight vyžadují instalaci přídavného softwaru, který nemusí být jednoduše dostupný na všech platformách, bylo od použití těchto technologií upuštěno. Nejlepší možností tedy bylo použít přímo HTML5, které nevyžaduje žádný přídavný software k tomu, aby výsledný program v počítači fungoval. Jedinou nutností je webový prohlížeč, který je ve všech běžných operačních systémech již v základu obsažen a v drtivé většině již dnes podporuje HTML5.

Součástí HTML5 jsou formulářové prvky, umožňující uživatelský vstup, jako jsou textová nebo číselná políčka, rozbalovací roletky a tlačítka, které se pro účely vznikajícího programu ideálně hodí. Pro zobrazení logických hradel a dalších součástek lze použít obrázky nebo grafické plátno, které je součástí HTML5.

Pro účely jednodušší orientace byl celý program rozčleněn do několika samostatných částí, které fungují nezávisle na sobě:

1. logické operace,
2. bitové operace,
3. úplná binární sčítačka,
4. sčítání, odčítání.

4.2 Implementace

Pro implementaci v HTML5 byly použity technologie a jazyky HTML, JavaScript a CSS. Jednou z možností bylo využít i framework jQuery, který usnadňuje práci s webovou stránkou. Kvůli udržení jednoduchosti implementace výsledného programu nebyl framework ve výsledné práci použit, protože žádná z jeho funkcí by nepřinesla zjednodušení.

Aby bylo možné program snadno přizpůsobovat na jednotlivé počítače, bylo potřeba vyřešit zobrazování schémat logických a bitových obvodů. Hlavní problém zde představuje rozlišení, to znamená, aby na výsledném počítači nebyl nákres příliš malý a nebo velký. Při použití obrázků musela být každá součástka ve zvláštním souboru a celková

velikost aplikace by se zvětšovala. Současně by obrázky měly fixní rozlišení a nebylo by tedy možné změnit velikost součástky na nákresu bez ztráty kvality. Jako nejlepší řešení se jevilo použití plátna, které je v HTML5 obsaženo.[16]

4.2.1 Vykreslování pomocí plátna

Technologie HTML5 je značkovací jazyk vycházející z jeho předchůdců, HTML a XHTML (případně SGML a XML). Popisuje sémantiku jednotlivých prvků a jejich chování na výsledné stránce. Pro potřeby vykreslování vektorové grafiky je v HTML5 obsažen prvek `<canvas>`, který představuje obdélníkovou oblast s možností vektorového kreslení. Koncept vykreslování je podobný jako u značkovacího jazyka SVG.[17]

Pro snažší vytváření schémat pomocí příslušných standardů vznikl oddělený soubor `drawing.js`, který je součástí této práce a výsledného programu. Všechny funkce v něm obsažené mají prefix `draw` a jako první parametr přijímají 2D kontext plátna. Následující zdrojový kód ukazuje nejjednodušší přístup pro práci s plátnem a získání 2D kontextu:

```
<!DOCTYPE>
<html>
  <head>
    <script>
      function getContext() {
        return document.querySelector("#canvas").getContext("2d");
      }
    </script>
   </head>
   <body>
    <canvas id="canvas" width="640" height="480" />
   </body>
</html>
```

Výpis 17: Ukázka práce s plátnem a získání kontextu

Dále je součástí souboru globální proměnná `drawConfig`, která umožňuje nastavit vlastnosti vykreslování, jako jsou barvy nebo typ a velikost písma.

Pro vykreslování spojů mezi součástkami slouží metody `drawWire`, `drawWireJoin` a `drawWireEnd`. Vstupy a výstupy obvodu lze vykreslit pomocí funkcí `drawInput` a `drawOutput`. Popis jednotlivých parametrů funkcí je jasně pochopitelný z jejich názvů.

Pro vykreslení součástky podle normy IEEE 91 slouží funkce `drawIEEE91`, která jako parametr přijímá obdélníkovou oblast, kam má být součástka vykreslena a parametry součástky. Pro jednoduchost jsou zde i funkce, které využívají a vykreslují již konkrétní součástky:

- `drawAND` pro logický součin,
- `drawOR` pro logický součet,
- `drawNAND` pro negovaný logický součin,
- `drawNOR` pro negovaný logický součet,

- `drawXOR` pro exkluzivní logický součet,
- `drawXNOR` pro exkluzivní negovaný logický součet.

Výjimkou je pak jediná unární operace NOT, která se vykresluje pomocí funkce `drawNOT`. Dále jsou v souboru obsaženy funkce pro vykreslování různých typů sčítaček.

4.2.2 Samotné části programu

Pomocí plátna a dalších prvků HTML5 jsou pak sestaveny samotné interaktivní animace. V části **Logické operace** má uživatel na výběr z rozevíracího seznamu, obsahující všechny v této práci popsané logické operace. Pod tímto seznamem se zobrazuje schématická součástka, představující logickou operaci podle normy IEEE 91. Na vstupu součástky jsou dvě tlačítka, která umožňují vybrat si hodnotu vstupu (buď 0 a nebo 1). Na výstupu se pak objeví aktuální hodnota podle zvolené operace. Současně je operace zobrazena v tabulce vedle součástky. Unární operace NOT je zobrazena zvlášť, protože má pouze jeden vstup, rovněž představovaný tlačítkem.

V druhé části práce nazvané **Bitové operace** má uživatel opět na výběr z rozevíracího seznamu. Dostupné možnosti jsou:

- bitový součin,
- bitový součet,
- bitová negace,
- exkluzivní bitový součet.

Uživatel zadává vstupní čísla pomocí číselných políček s rozsahem -128 až 255, který pokryje celý rozsah datového typu bajt, a to se znaménkem i bez znaménka. Současně se uživateli zobrazuje binární reprezentace zadaného čísla a výsledek bitové operace. Součástí je i zvláštní vstup pro bitový posun a rotaci.

Třetí část, **Úplná binární sčítačka**, obsahuje nákres sčítačky se třemi vstupy a dvěma výstupy. Jeden vstup a jeden výstup slouží k předávání bitu do vyššího řádu, aby bylo možné sčítačku řetězit. Schéma zobrazuje celý obvod sčítačky včetně mezivýsledků výpočtu.

Poslední část programu, **Sčítání, odčítání**, zobrazuje sčítačku složenou z osmi úplných binárních sčítaček. Sčítačka umí sečíst dvě osmibitová čísla a zobrazit výsledek součtu.

4.3 Testování a hodnocení

Výsledné interaktivní animace budou umístěny na jedné webové stránce a je možné je integrovat do LMS. V rámci testování bylo ověřeno, zda program funguje na různých prohlížečích a operačních systémech. Test probíhal na následujících konfiguracích:

- Windows 10, Google Chrome 49
- Windows 10, Microsoft Edge 25

- Windows 10, Microsoft Internet Explorer 11
- Windows 10, Mozilla Firefox 44
- Ubuntu 15.04, Google Chrome 49

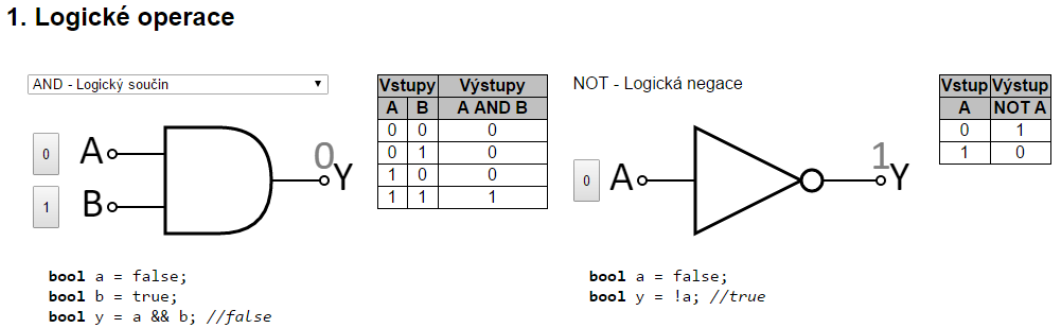
Program fungoval bez problémů na všech zmíněných konfiguracích, vyjma prohlížeče Microsoft Internet Explorer, jehož poslední verze byla vydána v roce 2013.[14] S ohledem na jeho malé zastoupení není nekompatibilita programu v IE 11 problémem.[15]

5 Studijní opory

Následující část práce popisuje, jak používat vzniklé interaktivní animace a jejich možnosti. Bližší informace k logickým a bitovým operacím pak popisuje uživatelská příručka, která je přílohou práce.

5.1 Logické operace

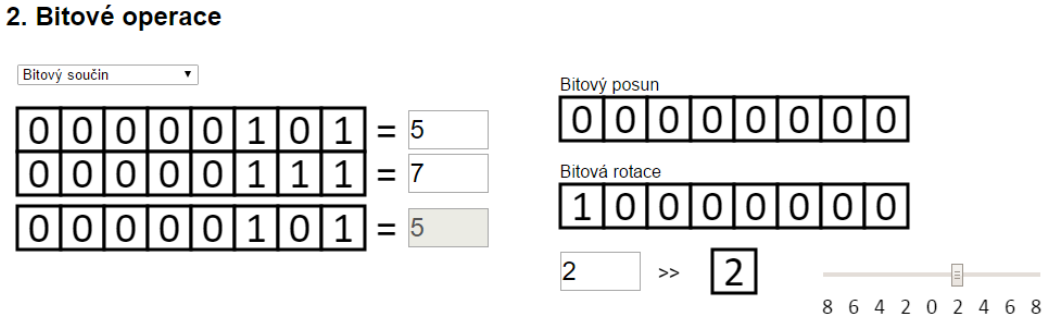
Rozevírací roletka umožňuje vybrat druh hradla a kliknutím na tlačítko u vstupu hradla je možné změnit vstupní hodnotu. Výsledek logické operace se zobrazuje u výstupu hradla. Způsob zpracování operace je viditelný v pravdivostní tabulce.



Obrázek 5.1: Animace logický operací

5.2 Bitové operace

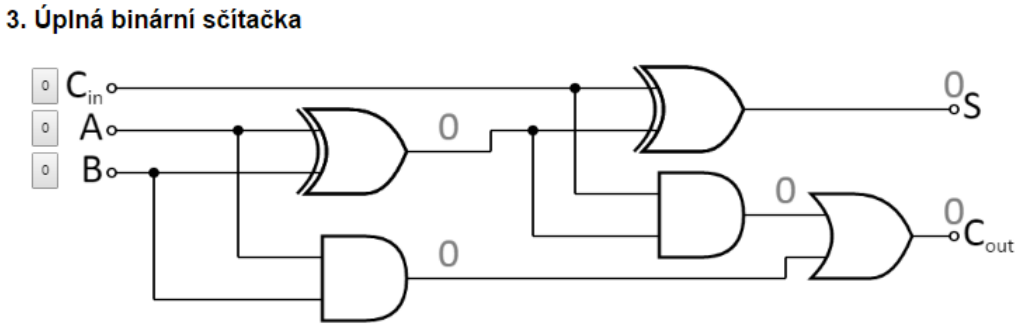
Pomocí rozevírací roletky lze vybrat jednu z možných bitových operací. Vstupy se zadávají skrze číselná políčka v desítkové soustavě. Animace je schopna zpracovat čísla v rozsahu jednoho bajtu a to, jak se znaménkem, tak i bez znaménka. Vstup pro bitový posun a rotaci je rovněž reprezentován číselným políčkem a operace je možná o osm bitů na obě strany. Posun i rotace se zadává souběžně pomocí posuvníku.



Obrázek 5.2: Animace bitových operací

5.3 Úplná binární sčítačka

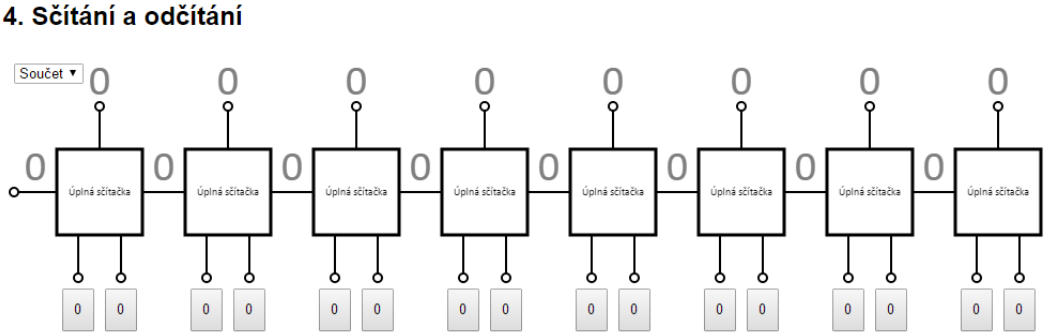
Vstup do úplné binární sčítačky se zadává pomocí tlačítek, které umožní změnu hodnoty na vstupu. Animace obvodu pak zobrazuje výsledné výstupy včetně průběžných výsledků. Tato část a následujíci část je pouze názorná ukázka, jak lze využít hradla k aritmetickým operacím.



Obrázek 5.3: Animace úplné binární sčítačky

5.4 Sčítání, odčítání

Rozevírací roletka umožňuje přepínání mezi sčítáním a odčítáním. Vstupy se zadávají pomocí tlačítek pod sčítačkou, přičemž levé vstupy představují jedno vstupní číslo a pravé vstupy druhé vstupní číslo. Obě čísla jsou tedy v rozsahu jednoho bajtu a výsledek je zobrazen na výstupech sčítačky.



Obrázek 5.4: Animace sčítání a odčítání

Závěr

Práce měla za úkol sumarizovat poznatky o logických operacích a hradlech, které je fyzicky realizují. Teorie se opírá o to, jak vznikly a proč jsou využívány dodnes. Druhou částí problematiky byly bitové operace a následně jsem se zaměřil na jejich porovnání a ukázal jejich rozdíly.

V průběhu své práce jsem se seznámil s elektronickým vzděláváním i z pohledu tvorby výukových materiálů. Zkoumal jsem jeho možnosti a jak tyto možnosti nejlépe uplatnit. Získal jsem nové znalosti na platformě HTML5, které jsem použil pro vytvoření výukového interaktivního programu. Pro jeho doplnění byly vytvořeny podpůrné studijní materiály. Vytvořil jsem tak základ, který lze i nadále rozšiřovat a přidávat do něj další části, což by mohlo být předmětem pro další rozvoj této práce. Rozšíření je schopen kdokoliv se základy jazyka HTML, CSS a JS. V průběhu tvorby jsem se přesvědčil dobrým výběrem použité technologie a vyvaroval se tak problémům, které by vznikaly při tvorbě jinou technologií.

Získané poznatky jsem se snažil zakomponovat do své práce, abych tak pomohl lepšímu porozumění tohoto problému. Získal jsem základy programovacího jazyka VHDL pro FPGA kit a seznámil se s implementací integrovaných obvodů softwarově. Poznal jsem také další vývojové nástroje od předního výrobce FPGA kitu Xilinx.

Výsledkem práce je tedy sada interaktivních animací pro výuku logických a bitových operací, vytvořených jako webová stránka. Součástí je i manuál k jejich použití.

Tomáš Schűhler

Literatura

[1] CHMELÍKOVÁ, Zdeňka a Jaroslav ZDRÁLEK. *Číslicové systémy I pro integrovanou výuku VUT a VŠB-TUO*. Ostrava: Vysoká škola báňská – technická univerzita Ostrava, 2014, i. ISBN 978-80-248-3649-2.

[2] KESL, Jan. *Elektronika III číslicová technika*. [Pbk. ed.] Praha: BEN - technická literatura, 2005, ii, 99 s. ISBN 80-7300-182-9

[3] ROTH, Charles H a Larry L KINNEY. *Fundamentals of logic design*. Stamford, CT: Cengage Learning, 2014, vii, 816 s. ISBN 1-133-62847-8.

[4] PRATT, Siofra. *The Beginner's Guide to Boolean Search Operators*[online]. 5.11.2015. Dostupné z: http://www.socialtalent.co/blog/the-beginners-guide-to-boolean-search-operators

[5] 91a-1991 - IEEE Graphic Symbols for Logic Functions (Includes IEEE Std 91A-1991 Supplement, and IEEE Std 91-1984). IEEE STANDARD, 1991.

[6] PETZOLD, Charles, Michael J SCHEARER a Frank THORNTON. *Code: the hidden language of computer hardware and software*.[Pbk. ed.]. Redmond, Wash: Microsoft Press, 2000, ix, 413 s. ISBN 978-073-5611-313.

[7] DIVIŠ, Zdeněk, Zdeňka CHMELÍKOVÁ a Jaroslav ZDRÁLEK. *Logické obvody.* 2. vyd. Ostrava: VŠB - Technická univerzita Ostrava, 2008. ISBN 978-80-248-1724-8

[8] *Assembler x86: Studijní text pro předmět: Strojově orientované jazyky. Poli.cs.vsb.cz* [online]. Ostrava: VŠB-TU, 2011 [cit. 2016-06-13]. Dostupné z: http://poli.cs.vsb.cz/edu/soj/down/soj-skripta.pdf

[9] PRATA, Stephen. *Mistrovství v C++.* 4., aktualiz. vyd. Přeložil Boris SOKOL. Brno: Computer Press, 2013. Bestseller (Computer Press). ISBN 978-80-251-3828-1.

[10]HEROUT, Pavel. *Učebnice jazyka Java.* 5. rozš. vyd. České Budějovice: Kopp nakladatelství, 2011. ISBN 978-80-7232-398-2.

[11]PÍSEK, Slavoj. *Delphi - začínáme programovat: podrobný průvodce začínajícího uživatele.* 2. upr. a aktualiz. vyd. Praha: Grada, 2002. ISBN 80-247-0547-8.

[12]Operace posuvu a rotace. *AmaPro: Internetové stránky určené pro studenty středních a vysokých odborných škol* [online]. [cit. 2016-06-14]. Dostupné z: http://www.amapro.cz/public/programovani/assembler/rotace.php

[13] *Basys 2^{TM} FPGA Board Reference Manual*[online]. 8.4.2016.
Dostupné z https://reference.digilentinc.com/_media/basys2:basys2_rm.pdf

[14] IEBlog *Internet Explorer Team Blog*[online]. c2016.
Dostupné z https://blogs.msdn.microsoft.com/ie/2013/11/07/ie11-for-windows-7-globally-available-for-consumers-and-businesses

[15] W3schools *The Internet Explorer Browser* [online]. c2016.
Dostupné z http://www.w3schools.com/browsers/browsers_explorer.asp

[16]*W3C A vocabulary and associated APIs for HTML and XHTML* [online]. revize 28.10.2014.
Dostupné z https://www.w3.org/TR/html5/

[17]*W3C The canvas element* [online]. revize 28.10.2014.
Dostupné z https://www.w3.org/TR/html5/scripting-1.html#the-canvas-element

A Výukové materiály k interaktivní animaci

A.1 První část animace - logické operace

K vyjádření libovolné logické funkce se v Booleově algebře používají jen tři základní funkce: logický součet, logický součin a logická negace. Těmito základními funkcemi můžeme vyjádřit libovolnou logickou operaci. Logické proměnné a logické funkce i konstanty nabývají v Booleové algebře jen dvou hodnot, takže vlastně tato algebra počítá ve dvojkové soustavě. Logické operace se znázorňují pomocí značek dle standardu IEEE Std 91a-1991.

A.1.1 Logický součin

Logický součin je logická funkce ve tvaru:

$$Y = A \cdot B$$

Kde Y s indexem značí výstup a symboly A a B jsou vstupními proměnnými a symbol $\cdot$ představuje operátor logického součinu. V některých literaturach je označován jako konjunkce, operace AND, logické násobení nebo logický součin. Logický člen, který realizujeme funkcí logického součinu, se označuje AND nebo v českém vyjádření čteme A.

A.1.2 Logický součet

Logický součet je logická funkce ve tvaru:

$$Y = A + B$$

Kde symbol $+$ představuje operátor logického součtu. Logický člen, který realizuje funkci logického součinu, se označuje OR v českém vyjádření čteme NEBO.

A.1.3 Logická negace

Nejjednodušší funkcí je logická negace. V algebraickém tvaru se vyjadřuje:

$$Y = \overline{A}$$

V některé literatuře se můžeme setkat s pojmem "A non". Hradlo se označuje NOT. Toto hradlo jako jediné provádí unární operaci, proto má jen jednu vstupní proměnnou.

A.1.4 Negovaný logický součin

Negací logického součinu je logická funkce ve tvaru:

$$Y = \overline{A \cdot B}$$

Tato funkce vzniká negací funkce AND, kde nadtržení znamená obrácení hodnoty výstupní proměnné. Negace je znázorněna kroužkem na výstupu hradla. Logický člen, který realizuje negovaný logický součin se označuje NAND. Alternativní názvy pro toto hradlo jsou negace konjunkce, operace NAND, negace AND, negace logického násobení, negovaný logický součin nebo Shefferova funkce.

A.1.5 Negovaný logický součet

Negací logického součtu je logická funkce ve tvaru:

$$Y = \overline{A + B}$$

Případně ekvivalentně zápisem:

$$Y = \overline{A} * \overline{B}$$

Tato funkce vzniká negací OR. Logický člen, který realizuje negovaný logický součet se označuje NOR. Alternativní názvy pro toto hradlo jsou negace disjunkce, operace NOR, negace OR, negace logického součtu nebo Peirceova funkce.

A.1.6 Exkluzivní logický součet

Takzvaný exkluzivní OR tedy XOR. Je poslední běžně využívané hradlo. Funkci to lze znázornit jako:

$$Y = \overline{A} * B + A * \overline{B}$$

Odpovídá matematickému součtu modulo 2. Alternativní názvy pro hradlo XOR je non-equivalence, exclusive OR.

A.1.7 Exkluzivní negovaný logický součet

Vychází z hradla XOR, ale výstup je negovaný. Alternativní názvy jsou ekvivalence, exclusive NOR.

A.1.8 Bitový součin

Nejzákladnější bitová operace je bitový součin. Máme-li čísla A, B a chceme provést bitový součin můžeme zapsat:

$$Y = A + B$$

Bitový součin se řídí podle tabulky 2.1 logické funkce AND. Pro binární součin, tak jako u aritmetické operace je potřeba brát v úvahu přenos do vyššího řádu z tohoto důvodu musí být vektor součtu větší rozsah o 1 řád. U bitového součinu přenos do vyššího nebo nižšího řádu neprobíhá.

A.2 Druhá část animace - bitové operace

Je třeba si uvědomit rozdíl mezi bitovými a binárními operacemi. Oba typy mohou být realizovány pomocí soustavy logických operací. Výstupem bitové operace je skupina bitů pozměněná bitovou operací. Implementace těchto operací je již v základních instrukčních sadách procesorů. O kterých je více zmíněno ve třetí kapitole. Na rozdíl od toho binární operace je totožná s aritmetickou pouze její realizace je ve dvojkové soustavě.

A.2.1 Bitový součin

Nejzákladnější bitová operace je bitový součin. Máme-li čísla A, B a chceme provést bitový součin můžeme zapsat pro každý bit:

$$Y = A + B$$

Bitový součin se řídí podle pravdivostní tabulky logické funkce AND pro každý bit. Pro binární součin, tak jako u aritmetické operace, je potřeba brát v úvahu přenos do vyššího řádu a z tohoto důvodu může být vektor součtu větší o jeden řád. U bitového součinu přenos do vyššího řádu neprobíhá.

A.2.2 Bitový součet

Druhá bitová operace je bitový součet. Pro každý bit platí:

$$Y = A \cdot B$$

Bitový součin se řídí podle pravdivostní tabulky 2.2 logické funkce OR.

A.2.3 Bitová negace

Unární operace, provádí logickou negaci každého bitu. Má pouze jeden operand. V případě nuly bude jedna a naopak. U bitové negace není důležité, zda je datový typ znaménkový nebo bezznaménkový. Při negaci bitů se nebere ohled na znaménko čísla, nejde tedy o ekvivalentní operaci se změnou znaménka reprezentovaného čísla.

A.2.4 Bitový posun

Bitový posun rozlišujeme na posun vlevo nebo posun vpravo o jeden či více bitů. Hodnoty jednotlivých bitů se posunou a chybějící bity jsou doplněny nulami. V případě bitového posunu doprava může být místo nul doplněna hodnota bitu nejvíce vlevo.

A.2.5 Bitová rotace

Bitové rotace se podobají bitovým posunům. Rozdíl je v tom, že bit který je při posunu vysouván z hodnoty ven, je při rotaci opět vkládán na opačném konci. Tato operace nebývá dostupná ve vyšších programovacích jazycích, je však ve instrukční sadě mnoha procesorů.

A.3 Třetí a čtvrtá část animace

Taty část jsou pouze názorné ukázky, jak lze využít hradla k aritmetickým operacím. Kombinací dvou polovičních sčítaček a logického součtu reprezentujeme úplnou sčítačku. Kde vstup C_{in} reprezentuje bit z nižšího řádu a výstup C_{out}, přenos bitu do vyššího řádu. Sčítačka umí stále sčítat pouze jednobitové operace, *viz 3. část animace* Zřetězením této sčítačky nám teprve vzniká více bitová sčítačka. Je zobrazena ve čtvrté části animace a lze ji plnohodnotně použít. Řetězením úplných sčítaček můžeme sčítat libovolné bitové vektory. Umožňuje sečíst dvě 8-mi bitová čísla s přenosem do vyššího řádu. Součet na výstupu může mít až 9 bitů, přičemž nejvyšší bit může být předán do vyššího řádu v případě řetězení více sčítaček. Hodnoty $A_0 - A_7$ představují vstup pro jedno binární číslo a $B_0 - B_7$ pro druhé. Součtem těchto čísel je pak číslo $S_7 - S_0$, přičemž nejvyšší devátý bit je označen jako C_{out} Vstup do úplné binární sčítačky se zadává pomocí tlačítek, které umožní změnu hodnoty na vstupu. Animace obvodu pak zobrazuje výsledné výstupy včetně průběžných výsledků. Rozevírací roletka umožňuje přepínání mezi sčítáním a odčítáním. Vstupy se zadávají pomocí tlačítek pod sčítačkou, přičemž levé vstupy představují jedno vstupní číslo a pravé vstupy druhé vstupní číslo. Obě čísla jsou tedy v rozsahu jednoho bajtu a výsledek je zobrazen na výstupech sčítačky.

B Obsah CD

Přiložené CD obsahuje:

- **interaktivní animace**: obsahuje *index.html*, *drawing.js* a *style.css*
- **materialy.pdf**: výukové materiály k interaktivní animaci
- **tex**: zdrojové kódy práce včetně obrázků
- **bp.pdf**: hotová práce v PDF